The ISP Column
A column on things Internet

Geoff Huston
November 2017

Three DNS articles:

1. DDOS and the DNS

The Mirai DDOS attack happened just over a year ago, on the 21st October 2016. The attack was certainly a major landmark in terms of the sorry history of "landmark" DDOS attacks in the Internet. It's up there with the Morris Worm of 1988, Slammer of 2002, Sapphine/Slammer of 2009 and of course Conficker in 2008. What made the Mirai attack so special? It was the largest we have seen so far, with an attack that amassed around 1Tb of attack traffic, which is a volume that creates not only a direct impact on the intended victim but wipes out much of the network infrastructure surrounding the attack point as well. Secondly, it used a bot army of co-opted web cams and other connected devices, which is perhaps a rather forbidding portent of the ugly side of the over-hyped Internet of Things. Thirdly, the target of the attack was aimed at the Internet's DNS infrastructure.

The first two aspects of this attack are just depressing, and it might well be that there is little we can do about them. However, the tragic hopelessness of the Internet of billions of Stupid Insecure Things is a story for another time, and the ever-increasing scale of attacks is perhaps also a constant with the same grim inevitability as Moore's Law. But perhaps we can do something about the DNS making itself such an easy target. Maybe we don't have to accept that the DNS has to be a victim of such attacks.

Mirai is by no means the first attack on the DNS infrastructure and it certainly will not be the last. If causing widespread Internet mayhem is your objective, then targeting the infrastructure of the DNS is certainly one way to achieve that. It's not even necessary to take out the root server system of the DNS to cause disruption. While there are many domain names to be served, there are a far smaller number of name servers, and of these there is an even smaller subset of these servers that serve so-called significant domain name. As was shown in the Mirai attack on the Dyn-operated name server infrastructure, a considered selection of a target of the attack can generate widespread impact to many online services. Just look at the impact list from the October 2016 Mirai attack (https://en.wikipedia.org/wiki/2016_Dyn_cyberattack).

There are few defences available to operators DNS server infrastructure. The attacks consist of otherwise perfectly legitimate queries. The name servers are put in the position of attempting to answer all queries that are presented to them, as there is no real way of determining what queries should be answered and what can safely be disregarded. As long as you can amass a large enough cloud of attack bots, and program them to perform DNS queries of random names within the target domain at a rate of a couple of queries per second, then the consequences can be relatively catastrophic for any name server. What we are seeing is that the DNS represents a point of vulnerability, and there are few enterprises who are in a position to mount a DNS name service that is invulnerable to such attacks as a matter of conventional online service provision.

When the environment gets toxic due to the risk of attacks we naturally retreat our valued resources to fortified castles. The thinking behind this move is that the castle builders have only a single product to offer: haven from attack, so they can specialise in defence while leaving the enterprise to get on with their actual online business, whatever that may be. However, as with any highly specialised activity, there are not a large number of these "castle builders" out there, and not a lot of castles with sufficiently thick and high walls. As the attacks mount in intensity, the requirements to increase the defensive capability of the castle escalates, so the cost of castle construction gets higher. The result of this escalation of the intensity of attack and the scale of the defence is one of aggregation and compaction in the castle builder industry as we retreat further to a smaller number of even more heavily fortified castles. In human terms this happened in Europe for many centuries as castles had to react to increasingly more powerful artillery. The artillery makers had to react to increasingly fortified castles, to there was a mutual spiral of escalation in the capability of both attack and defence.

The DNS is following an identical path. While it was feasible to set up an authoritative name server on conventional server hardware in days past, such approaches are tragically vulnerable in this day and age to even the most casual of attacks. If your service is of value to you and your clients, then you absolutely need to consider using a specialised DNS hosting service to host your name servers. But there are not a lot of these specialised server providers. The result is that there is increased fate sharing in the DNS as more DNS services are shifted onto just a few large scale server systems. It may seem paradoxical, but in the same way that castles drew the focus of attention of hostile forces, these small number of heavily fortified DNS service providers are drawing an undue share of attraction from attackers, while at the same time they carry a large number of significant services.

Giovane Moura of SIDN Labs reported in the IEPG meeting at IETF 100 (http://bit.ly/2jDH8up) on some early findings about what he terms as "oversharing" in the DNS. To what extent are we placing our DNS server resources into a small set of locations, and if so then does this represent a new risk factor to the DNS? For example, as part of his findings, he observed that the authoritative name servers operated by Ausregistry (which is owned these days by Neustar) now host some 363 top level domain names.

It's entirely reasonable to expect that once a castle builder has spent the money to build a large castle to try and sell apartments in the castle to as many tenants as possible. But it's also entirely reasonable to observe that an attack on the castle damages all tenants, and the greater the number of tenants the greater the likelihood that one of more of these tenants will be the subject of an attack. The more people who take shelter in a castle the more the castle represents an attractive point of attack to a hostile attacker

As with many other areas of Internet infrastructure, economies of scale are present, and the greater the scale of the facility or service, the more likely the service operator can realise efficiencies that allow the operator to offer a price competitive service. The question raised in the presentation is when do the economies of scale of sharing a common any cast infrastructure platform top over to become a risk factor of oversharing. As we've already observed, it's not possible to totally impregnable infrastructure using this approach of fortified servers. The size of today's attacks can readily overwhelm anything we can build, so the risks that this "oversharing" exposes are very real risks.

> The entire approach of loading critical assets into castles and using the castle's defences to withstand attack, strains any rational credibility at the point when attacks can be mounted that can overwhelm all known points of defence of the castle.
>
> The introduction of gunpowder and subsequent refinement of artillery starting in Europe from the 14th century s probably why we stopped building castles many years ago! By the 18th century, after building some 100,000 castles across western Europe, castle building turned from being a military activity to an amusement inspired by a romantic revival for the wealthy. Perhaps the best example of this is Ludwig II of Bavaria's Neuschwanstein castle, constructed in the late 19th century

It seems that we've come to the point in the evolution of the DNS infrastructure where what we have working on for some years is no longer effective, and the promises from the dedicated server operators of greater resiliency and survivability when under malicious attack seem to lack credibility.

What can we do about this particular problem?

In my mind, the answer comes, unexpectedly, from DNSSEC.

Firstly, lets understand how malicious agents mount these DNS DOS attacks. Repeatedly asking the same question of the DNS through existing recursive resolvers will not generate an attack on the authoritative servers. The caches that sit within these recursive resolvers will match the query to one that was recently answered, and just respond to the query with the cached response. This is not the way to overload an authoritative name server. The attacker needs to generate as unique name for each query, and in that case the recursive resolver will pass the query onward to the authoritative name server, as the name has not been seen before.

How can we defend against random name attacks?

Here the behaviour of DNSSEC helps. DNSSEC responses to queries for non-existent names require a signed response that "proves" that the name is not in the zone. But DNSSEC can only sign names that exist in the zone, so the response uses a zone "range" response, providing the name before the queries name and the name after the queried name, using either lexicographic ordering (NSEC) or ordering of the hash of the names (NSEC3). The undertaking from the authoritative server is that no name exists between these two names within the scope of that ordering of the zone's contents.

> For example, if we were to use the Oxford Dictionary of the English language, and look up the nonsense word "liverx", the dictionary can tell us that there are no valid English words between "liverwort" and "livery".

> Because DNSSEC can only sign names that exist in a zone, if we have the hypothetical case where the Oxford Dictionary was mapped into the DNS, and signed using DNSSEC, then the equivalent of a lookup for "liverx" would result in a NXDOMAIN response referencing the signed NSEC record for "liverwort" which notes that "livery" is the subsequent name in the ordered list of all names in this zone

For example, the root zone of the DNS contains the following records:

```
am.      86400   IN  NSEC    americanexpress. NS DS RRSIG NSEC
am.      86400   IN  RRSIG   NSEC 8 1 86400 2017 …
```

When a DNSSEC_enabled resolver queries for a non-existent name in this range, such as "ama" the NXDOMAIN response includes this NSEC record and its associated signature. This NSEC response is telling us so much more than just the fact that the label "ama" is not a delegated label in the root zone. It's telling us that all labels that lie between "am" and "americanexpress" are not defined in the root zone, and of course there are a very large set of such non-existent labels. However, for each such label, if it were to be queried, the NXDOMAIN response would include exactly the same NSEC records. If the recursive resolver were to cache these NSEC responses it could use its cache to respond to such queries, either providing the entire NSEC records set if the query included the DNSSEC OK flag setting, or a simple NXDOMAIN otherwise.

The consequence is that we now have a very efficient way of caching the fact of nonexistence of domain names.

What happens to an attack using random name queries when the zone is signed and the recursive resolvers are caching NSEC records? A recursive resolver will not pass these queries to the authoritative server, but will use its own local cache to respond once it loads its cache. So, if recursive resolvers are set up to perform DNSSEC validation, and if a zone is DNSSEC-signed, then random name attacks on authoritative name servers will be absorbed by the recursive resolvers.

This has the potential to change the perception of DNSSEC. Signing the DNS has always been a difficult proposition. The cache poisoning attacks that DNSSEC can prevent are somewhat esoteric, and many zone administrators see little direct benefit in signing their zones. At the same time while so few zones are signed, then the value proposition for recursive resolvers to perform DNSSEC validation was similarly weak.

However, DNSSEC and NSEC caching has a far broader area of application, namely in mitigating the common form of attack on authoritative name servers, namely the random name attack. If resolvers perform DNSSEC validation and NSEC caching then they can operate a more efficient cache for signed zones, so they can answer all queries from their cache, rather than passing all such queries to the authoritative name servers. At the same time the authoritative name servers would see a major drop in query traffic, and in particular, a very significant drop in random name attack traffic.

It's now clear that DNSSEC can help us here. Resolvers should attempt to validate DNSSEC-signed responses by default. And if the response is NXDOMAIN, the resolvers should cache the NSEC response and reuse that response for the cache lifetime of the NSEC record, as per RFC8198. If zone administrators really want to secure their name service, then just serving their name from within an anycast castle is not enough. What we need is for the millions of recursive resolvers to be able to help deflect random name attacks. But perhaps this is not such an impossibly distant goal as It may have seemed in the past. There are a very small number of providers of DNS resolution code, and if these

providers were to include support for RFC8198 Aggressive NSEC caching in their code and switch it on in the default distribution, then the uptake of this rather critical measure would be much quicker.

I find that I agree with the conclusions drawn in Giovane Moura's IEPG presentation that oversharing in the top level domain space across a small number of specialised servers is a distinct risk for the resilience of the DNS. But where to go from here is the question. Constructing more heavily fortified anycast DNS server castles adds cost to the DNS without tangible benefit to anyone, so that path does not particularly appeal to me.

It's often the case in our current Internet that everyone wants to solve a problem at all levels of the protocol stack simultaneously. We want to improve the performance speed of applications by shaving time off the DNS request, and naturally the DNS providers leap on this and paint a picture where their anycast cloud offers fast resolution performance from all of their anycast points "just in case" the application that uses these DNS names happens to be running in the served region. A lot of information is being pushed to a lot of servers a lot of the time just in case a resolver asks to resolve that name. It seems like an extravagant solution to me.

I wonder if perhaps we are just not looking at this problem in a way that leads to different ways to address the issue. The role of authoritative name servers in the DNS, from the root downward, is not in fact to answer all queries generated by end users. The strength of the DNS lies in its caching ability, so that recursive resolvers handle the bulk of the query load. Sure, first lookups are slower than subsequent lookups because of cache priming operations, but one a name is served from the local cache it's as fast as your local resolver can make it. Authoritative servers answer cache misses from recursive resolvers. That's their role. That's the entirety of their role.

So this leads to a different response if you want the DNS resolution for your name to be as fast as possible. Set your zone's cache lifetime to be long. Once the resolution outcome is loaded into the local recursive resolver's cache, then the authoritative server is just not a factor in performance until the cache expires. So perhaps if the application is not attuned to make efficient use of the DNS you can also assist the application by using long cache TTLs for your stable DNS data. Yes, this is a tradeoff of timelieness to change and performance of resolution, and many DNS administrators will pick different points along a spectrum of cacheability vs dynamic control. But once more the beauty of the DNS is that this parameter can be set for each zone, so each zone administrator can tune their zone to behave according to their requirements. One size does not have to fit all.

It is also an option to tune the application. An application that initiates a DNS resolution as a background task well in advance of the time of object retrieval will eliminate much of the perceived latency of the application. Applications that make extensive use of diversely named objects are susceptible to performance improvement through using parallelism of these DNS requests as well as background pre-fetching.

It took some hundreds of years, but Europe eventually reacted to the introduction of gunpowder and artillery by recognising that they simply could not build castles large enough to defend against any conceivable attack. So they stopped.

I hope it does not take us the same amount of time to understand that building ever more massively fortified and over-provisioned servers is simply a tactic for today, not a strategy for tomorrow.

Author

Geoff Huston B.Sc., M.Sc., is the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region.

www.potaroo.net

Disclaimer

The above views do not necessarily represent the views or positions of the Asia Pacific Network Information Centre.